	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
1	BRS	L1	0	interlock same speculative same out-of-oder adj execution same load adj instruction	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 10:35	
2	BRS	L2	0	interlock same speculative same out-of-oder adj execution and load adj instruction	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:37	
3	BRS	L3	0	interlock same speculative same (out-of-oder adj execution) and (load adj instruction)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:38	
4	BRS	L4	0	speculative same (out-of-oder adj execution) and (load adj instruction)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:38	
5	BRS	L5	748	(speculative same execution) and (load adj instruction)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:38	
6	BRS	L6	0	5 and out-of-oder adj execution	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:39	

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
7	BRS	L7	0	5 and out-of-oder	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:39	
8	BRS	L8	273	5 and out-of-order adj execution	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:40	
9	BRS	L9	1	interlock same speculative same out-of-order adj execution same load adj instruction	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:40	
10	BRS	L10	4	8 AND priority adj3 level	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 10:37	
11	BRS	L11	269	8 not 10	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:45	
12	BRS	L12	0	11 and (youger or freshest)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 10:39	

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
13	BRS	L13	103	11 and source adj3 register	US-PGPUB; USPAT; EP ; JPO; DERWENT; IBM_TDB	2004/12/06 08:46	
14	BRS	L14	0	13 and non-load adj3 annex	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:47	
15	BRS	L15	1	13 and non-load adj3 instruction	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:49	
16	BRS	L16	14	13 and scoreboard	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:54	
17	IS&R	L17	2	("5878245").PN.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:54	
18	IS&R	L18	1695	(712/225,217,218,219,215).CCLS.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:55	

	Type	L #	Hits	Search Text	DBs	Time Stamp	C mments
19	BRS	L19	1244	18 and load	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:55	
20	BRS	L20	423	19 and priority	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 08:55	
21	BRS	L21	397	20 not 13	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 09:33	
22	BRS	L23	2	13 and by-pass	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 09:34	
23	BRS	L22	12	21 and by-pass	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 09:35	
24	BRS	L24	1	interlock and speculative same out-of-order adj execution same load adj instruction	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 10:36	

	Type	L #	Hits	Search Text	DBs	Tim Stamp	Comments
25	BRS	L25	17	interlock and speculative and out-of-order adj execution and load adj instruction	US-PGPUB; USPAT; EP ; JPO; DERWENT; IBM_TDB	2004/12/06 10:36	
26	BRS	L26	1	25 AND priority adj3 level	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 10:37	
27	BRS	L27	7	25 AND priority	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 10:37	
28	BRS	L28	1	27 and (youger or freshest)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 10:40	
29	BRS	L29	1	25 and (youger or freshest)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 10:40	
30	BRS	L30	3	25 and (younger or freshest)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/06 10:40	

Terms used
interlock and **speculative** and **out** **of** **order** and **execution** and **load instruction** **anf** **priority** and **younger** and **register** and **annex**

Found **1,924** of **147,060**

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Register integration: a simple and efficient implementation of squash reuse

Amir Roth, Gurindar S. Sohi

December 2000 **Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture**

Full text available: pdf(154.98 KB) ps(573.81 KB) Publisher Site

Additional Information: full citation, references, citings, index terms

2 A survey of processors with explicit multithreading

Theo Ungerer, Borut Robič, Jurij Šilc

March 2003 **ACM Computing Surveys (CSUR)**, Volume 35 Issue 1

Full text available: pdf(920.16 KB)

Additional Information: full citation, abstract, references, index terms

Hardware multithreading is becoming a generally applied technique in the next generation of microprocessors. Several multithreaded processors are announced by industry or already into production in the areas of high-performance microprocessors, media, and network processors.A multithreaded processor is able to pursue two or more threads of control in parallel within the processor pipeline. The contexts of two or more threads of control are often stored in separate on-chip register sets. Unused i ...

Keywords: Blocked multithreading, interleaved multithreading, simultaneous multithreading

3 Data prefetching by dependence graph precomputation

Murali Annavaram, Jignesh M. Patel, Edward S. Davidson

May 2001 **ACM SIGARCH Computer Architecture News , Proceedings of the 28th annual international symposium on Computer architecture**, Volume 29 Issue 2

Full text available: pdf(909.40 KB)

Additional Information: full citation, abstract, references, citings, index terms

Data cache misses reduce the performance of wide-issue processors by stalling the data supply to the processor. Prefetching data by predicting the miss address is one way to tolerate the cache miss latencies. But current applications with irregular access patterns make it difficult to accurately predict the address sufficiently early to mask large cache miss latencies. This paper explores an alternative to predicting prefetch addresses, namely precomputing them. The Dependence Graph Pr ...

4 On pipelining dynamic instruction scheduling logic

Jared Stark, Mary D. Brown, Yale N. Patt

December 2000 **Proceedings of the 33rd annual ACM/IEEE international symposium n Microarchitecture**

Full text available: pdf(128.82 KB) ps(543.84 KB) Publisher Site

Additional Information: full citation, references, citings, index terms

5 Macro-op Scheduling: Relaxing Scheduling Loop Constraints

Ilhyun Kim, Mikko H. Lipasti

December 2003 **Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture**

Full text available: pdf(356.70 KB) Publisher Site

Additional Information: full citation, abstract, index terms

Ensuring back-to-back execution of dependent instructionsin a conventional out-of-order processor requiresscheduling logic that wakes up and selects instructions atthe same rate as they are executed. To sustain high performance,integer ALU instructions typically have single-cyclelatency, consequently requiring scheduling logic withthe same single-cycle latency. Prior proposals have advocatedthe use of speculation in either the wakeup or selectphases to enable pipelining of scheduling logic to ac ...

6 Implementation trade-offs in using a restricted data flow architecture in a high performance RISC microprocessor

M. Simone, A. Essen, A. Ike, A. Krishnamoorthy, T. Maruyama, N. Patkar, M. Ramaswami, M. Shebanow, V. Thirumalaiswamy, D. Tovey

May 1995 **ACM SIGARCH Computer Architecture News , Proceedings of the 22nd annual international symposium on Computer architecture**, Volume 23 Issue 2

Full text available: pdf(1.04 MB)

Additional Information: full citation, abstract, references, citings, index terms

The implementation of a superscalar, speculative execution SPARC-V9 microprocessor incorporating Restricted Data Flow principles required many design trade-offs. Consideration was given to both performance and cost. Performance is largely a function of cycle time and instructions executed per cycle while cost is primarily a function of die area. Here we describe our Restricted Data Flow implementation and the means with which we arrived at its configuration. Future semiconductor technology advan ...

7 Extended Split-Issue: Enabling Flexibility in the Hardware Implementation of NUAL VLIW DSPs

March 2004 **ACM SIGARCH Computer Architecture News , Proceedings of the 31st annual international symposium on Computer architecture**, Volume 32 Issue 2

Full text available: pdf(281.39 KB) Additional Information: full citation, abstract

VLIW architecture based DSPs have become widespread due to thecombined benefits of simple hardware and compiler-extractedinstruction-level parallelism. However, the VLIW instruction setarchitecture and its hardware implementation are tightly coupled,especially so for Non-Unit Assumed Latency (NUAL) VLIWs. Theproblem of object code compatibility across processors having differentnumbers of functional units or hardware latencies has beenthe Achilles' heel of this otherwise powerful architecture. I ...

8 Continuous program optimization: A case study

Thomas Kistler, Michael Franz

July 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 4

Full text available: pdf(877.67 KB)

Additional Information: full citation, abstract, references, index terms, review

Much of the software in everyday operation is not making optimal use of the hardware on which it actually runs. Among the reasons for this discrepancy are hardware/software mismatches, modularization overheads introduced by software engineering considerations, and the inability of systems to adapt to users' behaviors.A solution to these problems is to delay code generation until load time. This is the earliest point at which a piece of software can be fine-tuned to the actual capabilities of the ...

Keywords: Dynamic code generation, continuous program optimization, dynamic reoptimization

9 A fill-unit approach to multiple instruction issue 

Manoj Franklin, Mark Smotherman

November 1994 **Proceedings of the 27th annual international symposium on Microarchitecture**

Full text available: pdf(992.94 KB) Additional Information: full citation, abstract, references, citings, index terms

Multiple issue of instructions occurs in superscalar and VLIW machines. This paper investigates a third type of machine design, which combines the advantages of code compatibility as in superscalars and the absence of complex dependency-checking logic from the decoder as in VLIW. In this design, a stream of scalar instructions is executed by the hardware and is simultaneously compacted into VLIW-type instructions, which are then stored in a structure called a shadow cache. When a shadow cac ...

Keywords: VLIW, instruction-level parallelism, multiple operation issue, superscalar

10 Zero-cycle loads: microarchitecture support for reducing load latency 

Todd M. Austin, Gurindar S. Sohi

December 1995 **Proceedings of the 28th annual international symposium on Microarchitecture**

Full text available: pdf(1.35 MB) Additional Information: full citation, references, citings, index terms

11 Software support for speculative loads 

Anne Rogers, Kai Li

September 1992 **ACM SIGPLAN Notices , Proceedings of the fifth international conference on Architectural support for programming languages and operating systems**, Volume 27 Issue 9

Full text available: pdf(1.33 MB) Additional Information: full citation, references, citings, index terms

12 Enhancing software reliability with speculative threads 

Jeffrey Oplinger, Monica S. Lam

October 2002 **Proceedings of the 10th international conference on Architectural support for programming languages and operating systems**, Volume 37 , 36 , 30 Issue 10 , 5 , 5

Full text available: pdf(1.47 MB) Additional Information: full citation, abstract, references, citings

This paper advocates the use of a monitor-and-recover programming paradigm to enhance the reliability of software, and proposes an architectural design that allows software and hardware to cooperate in making this paradigm more efficient and easier to program.We propose that programmers write monitoring functions assuming simple sequential execution semantics. Our architecture speeds up the computation by executing the monitoring functions speculatively in parallel with the main computation. For ...

13 Superscalar design: Cherry: checkpointed early resource recycling in out-of-order microprocessors 

José F. Martínez, Jose Renau, Michael C. Huang, Milos Prvulovic, Josep Torrellas
November 2002 **Proceedings f the 35th annual ACM/IEEE international symposium on Microarchitecture**

Full text available: pdf(1.40 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms, review

This paper presents *CHeckpointed Early Resource RecYcling (Cherry)*, a hybrid mode of execution based on ROB and checkpointing that decouples resource recycling and instruction retirement. Resources are recycled early, resulting in a more efficient utilization. Cherry relies on state checkpointing and rollback to service exceptions for instructions whose resources have been recycled. Cherry leverages the ROB to (1) not require in-order execution as a fallback mechanism, (2) allow memory re ...

14 Speculative execution via address prediction and data prefetching
José González, Antonio González
July 1997 **Proceedings of the 11th international conference on Supercomputing**

Full text available: pdf(1.33 MB) Additional Information: full citation, references, citings, index terms

15 Sentinel scheduling: a model for compiler-controlled speculative execution
Scott A. Mahlke, William Y. Chen, Roger A. Bringmann, Richard E. Hank, Wen-Mei W. Hwu, B. Ramakrishna Rau, Michael S. Schlansker
November 1993 **ACM Transactions on Computer Systems (TOCS)**, Volume 11 Issue 4

Full text available: pdf(2.26 MB) Additional Information: full citation, abstract, references, citings, index terms

Speculative execution is an important source of parallelism for VLIW and superscalar processors. A serious challenge with compiler-controlled speculative execution is to efficiently handle exceptions for speculative instructions. In this article, a set of architectural features and compile-time scheduling support collectively referred to as sentinel scheduling is introduced. Sentinel scheduling provides an effective framework for both compiler-controlled speculative executi ...

Keywords: VIIW processor, exception detection, exception recovery, instruction scheduling, instruction-level parallelism, speculative execution, superscalar processor

16 A novel renaming scheme to exploit value temporal locality through physical register reuse and unification
Stephen Jourdan, Ronny Ronen, Michael Bekerman, Bishara Shomar, Adi Yoaz
November 1998 **Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture**

Full text available: pdf(2.41 MB) Additional Information: full citation, references, citings, index terms

Keywords: dependency redirection, physical register reuse, register and memory renaming, result reuse, value temporal locality

17 Post-pass binary adaptation for software-based speculative precomputation
Steve S.W. Liao, Perry H. Wang, Hong Wang, Gerolf Hoflehner, Daniel Lavery, John P. Shen
May 2002 **ACM SIGPLAN Notices , Proceedings f the ACM SIGPLAN 2002 Conference on Programming language design and implementation**, Volume 37 Issue 5

Full text available: pdf(330.60 KB) Additional Information: full citation, abstract, references, citings, index terms

Recently, a number of thread-based prefetching techniques have been proposed. These techniques aim at improving the latency of single-threaded applications by leveraging

multithreading resources to perform memory prefetching via speculative prefetch threads. Software-based speculative precomputation (SSP) is one such technique, proposed for multithreaded Itanium models. SSP does not require expensive hardware support-instead it relies on the compiler to adapt binaries to perform prefetching on o ...

Keywords: chaining speculative precomputation, delay minimization, dependence reduction, long-range thread-based prefetching, loop rotation, pointer, post-pass, prediction, scheduling, slack, slicing, speculation, triggering

18 Speculative execution: Dynamic memory instruction bypassing

Daniel Ortega, Eduard Ayguadé, Mateo Valero

June 2003 **Proceedings of the 17th annual international conference on Supercomputing**

Full text available: pdf(216.24 KB) Additional Information: full citation, abstract, references, index terms

Reducing the latency of *load* instructions is among the most crucial aspects to achieve performance for current and future microarchitectures. Deep pipelining makes L1 caches appear farther than 1 cycle, thus impacting *load-to-use* latency, even if these instructions hit in cache. In this paper we present a novel dynamic mechanism aimed at overcoming *load-to-use* latency. Our mechanism dynamically detects relations between *address producing instructions*. and the *loads* ...

Keywords: *on-chip memory management, superscalar processors*

19 Fast out-of-order processor simulation using memoization

Eric Schnarr, James R. Larus

October 1998 **Proceedings of the eighth international conference on Architectural support for programming languages and operating systems**, Volume 32 , 33 Issue 5 , 11

Full text available: pdf(1.43 MB) Additional Information: full citation, abstract, references, citings, index terms

Our new out-of-order processor simulatol; FastSim, uses two innovations to speed up simulation 8--15 times (vs. Wisconsin SimpleScalar) with no loss in simulation accuracy. First, FastSim uses speculative direct-execution to accelerate the functional emulation of speculatively executed program code. Second, it uses a variation on memoization---a well-known technique in programming language implementation---to cache microarchitecture states and the resulting simulator actions, and then "fast forw ...

Keywords: direct-execution, memoization, out-of-order processor simulation

20 Predictive techniques for aggressive load speculation

Glenn Reinman, Brad Calder

November 1998 **Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture**

Full text available: pdf(1.94 MB) Additional Information: full citation, references, citings, index terms

h c g e cf c